QR-type iterations for eigenvalue computations based on factorizations of unitary matrices in rotations

Raf Vandebril

Report TW 577, September 2010

Katholieke Universiteit Leuven
Department of Computer Science
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

QR-type iterations for eigenvalue computations based on factorizations of unitary matrices in rotations

Raf Vandebril

Report TW 577, September 2010

Department of Computer Science, K.U.Leuven

Abstract

The QR-algorithm is a renowned method for computing all eigenvalues of an arbitrary matrix. A preliminary unitary similarity transformation to Hessenberg form is indispensible for keeping the computational complexity of the QR-algorithm applied on the resulting Hessenberg matrix under control.

The unitary factor Q in the QR-factorization of the Hessenberg matrix $H = QR$ is composed of $n-1$ rotations ordered from top to bottom. The ordering of these $n-1$ rotations is precisely determined by the Hessenberg structure and uniquely specifies the unitary matrix Q.

In this article it will be shown that for any matrix there exists a unitary similar matrix $A = QR$, where any type of ordering of the $n-1$ rotations in the factorization of the unitary matrix Q are admitted. Classic examples of rotational factorizations of a unitary matrix Q are sequences from bottom to top (lower unitary Hessenberg) or for instance the CMV-decomposition of a pentadiagonal unitary matrix. An important consequence of these alternative formats is the loss of the Hessenberg structure. The resulting matrix is, however, still representable by the same order of parameters. A constructive procedure to compute the unitary similar matrix in factored form $A = QR$, as well as proof of unicity of the reduction are given.

Based on the factorization of the unitary matrix Q in $n-1$ rotations an iterative procedure for computing the eigenvalues is presented. The QR-like iteration takes advantage of the specific ordering of the rotations and based on the unicity of the rotational factorization an implicit version is designed. The computational cost of the implicit version is again comparable to the cost of the QR-algorithm for Hessenberg matrices.

A first numerical experiment investigates to Ritz-value convergence behavior of some variants of the unitary similarity transformation. A second test investigates the speed of convergence and the accuracy of the new iterative method for some variants of rotational factorizations.

Keywords : unitary similarity transformations, QR-like algorithms, Givens rotations, patterns of rotations, eigenvalues

MSC : Primary : 15A18, Secondary : 15A21, 15A23

QR-TYPE ITERATIONS FOR EIGENVALUE COMPUTATIONS BASED ON FACTORIZATIONS OF UNITARY MATRICES IN ROTATIONS *

RAF VANDEBRIL†

Abstract. The *QR*-algorithm is a renowned method for computing all eigenvalues of an arbitrary matrix. A preliminary unitary similarity transformation to Hessenberg form is indispensible for keeping the computational complexity of the *QR*-algorithm applied on the resulting Hessenberg matrix under control.

The unitary factor Q in the *QR*-factorization of the Hessenberg matrix $H = QR$ is composed of $n-1$ rotations ordered from top to bottom. The ordering of these $n-1$ rotations is precisely determined by the Hessenberg structure and uniquely specifies the unitary matrix Q.

In this article it will be shown that for any matrix there exists a unitary similar matrix $A = QR$, where any type of ordering of the $n-1$ rotations in the factorization of the unitary matrix Q are admitted. Classic examples of rotational factorizations of a unitary matrix Q are sequences from bottom to top (lower unitary Hessenberg) or for instance the *CMV*-decomposition of a pentadiagonal unitary matrix. An important consequence of these alternative formats is the loss of the Hessenberg structure. The resulting matrix is, however, still representable by the same order of parameters. A constructive procedure to compute the unitary similar matrix in factored form $A = QR$, as well as proof of unicity of the reduction are given.

Based on the factorization of the unitary matrix Q in $n-1$ rotations an iterative procedure for computing the eigenvalues is presented. The *QR*-like iteration takes advantage of the specific ordering of the rotations and based on the unicity of the rotational factorization an implicit version is designed. The computational cost of the implicit version is again comparable to the cost of the *QR*-algorithm for Hessenberg matrices.

A first numerical experiment investigates to Ritz-value convergence behavior of some variants of the unitary similarity transformation. A second test investigates the speed of convergence and the accuracy of the new iterative method for some variants of rotational factorizations.

Key words. unitary similarity transformations, *QR*-like algorithms, Givens rotations, patterns of rotations, eigenvalues

AMS subject classifications. 15A18, 15A21, 15A23

1. Introduction. Among the so-called direct methods for eigenvalue computations, the *QR*-method is the most popular and most used one for determining the eigenvalues of nonsymmetric matrices. An extensive list of publications is devoted to this favorable technique: introductions appear in almost all numerical linear algebra textbooks [14, 18, 24], more sophisticated books [22, 40, 45], detailed studies of the convergence behavior linking the *QR*-algorithm with subspace iteration [36, 41, 43], articles about particular characteristics such as balancing, bulge chasing, maintaining well-focused shifts [38, 42, 44] and even very recently several improvements to achieve speed up and maintaining high accuracy were proposed [7, 8]. Also a wide variety of publications, both theoretical [12, 13] as more practical, is related to studies on how to adapt the *QR*-algorithm to make it suitable for particular matrix structures such as quasiseparable [16], semiseparable (plus diagonal) [30, 33] unitary plus low rank [6, 26], Hermitian plus low rank [28], companion, comrade, Hamiltonian [17, 39] and many other matrices.

Important, however, for applying a *QR*-algorithm is to perform a preprocessing step, thereby transforming the involved matrix to a more economical format. This format is assumed to have some prominent features: it admits a cheaper *QR*-step; generically less memory is needed to store the matrix and important is that the economical format does not get lost after a *QR*-iteration step. Well-known economical formats are the tridiagonal form (for a Hermitian matrix) and the Hessenberg matrix (for a nonsymmetric matrix). The original matrix is brought via unitary similarity transformations to this shape retaining thereby the eigenvalues [14, 18, 40]. Some other less widespread formats are related to rank structured matrices [15, 25]. Also reduction algorithms tuned for specific matrix cases exist [3, 21].

In this article both the unitary similarity transformation and the *QR*-algorithm are reconsidered generalizing thereby the results for Hessenberg and Hessenberg-like (inverses of Hessenberg) matrices. Examining the Q-factor in the *QR*-factorization of Hessenberg(-like) matrices, it is observed that it can be factored by using $n-1$ Givens rotations. These $n-1$ rotations are in fact the minimal number that can be achieved by a unitary similarity transformation on an arbitrary unitary matrix. The resulting unitary matrix

*The author has a grant as "Postdoctoraal Onderzoeker" from the Fund for Scientific Research–Flanders (Belgium). The research was also partially supported by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office, Belgian Network DYSCO (Dynamical Systems, Control, and Optimization).

†K.U.Leuven, Dept. of Computer Science, 3001 Leuven (Heverlee), Belgium. (raf.vandebril@cs.kuleuven.ac.be).

is referred to as being in compressed format. For a Hessenberg matrix this compressed format is build by $n-1$ rotations in descending order, a Hessenberg-like matrix has $n-1$ rotations in ascending order. Studies [1, 4, 9, 10, 23, 37] relating unitary matrices with orthogonal Laurent polynomials on the unit circle reveal, however, the existence of a wide variety of possible compressed factorizations of a unitary matrix. Instead of only considering ascending or descending sequences of rotations, an arbitrary number of direction changes can be taken. For example, the nowadays popular *CMV*-decomposition presents a compressed factorization of a unitary matrix, changing direction for each of the $n-1$ rotations.

Based on a pre-specified pattern that the compressed unitary matrix should satisfy a new algorithm for executing the similarity transformation is presented leading to a *QR*-factorization $\hat{A} = QR$, with Q in desired format. Globally this method has the same computational complexity as the transformation to Hessenberg form. Both the Hessenberg and Hessenberg-like case are contained as instances of this generic reduction method. Another example includes a resulting matrix *QR*-factored, where Q is decomposed in $n-1$ rotations according to the *CMV*-pattern. A proof of unicity of the reduction procedure based on Krylov sequences is included.

This almost unlimited number of patterns that can be used to reduce a matrix to are then used to develop a *QR*-like algorithm. Both an explicit as well as an implicit version of a *QR*-like version suitable for the new structures will be presented. It will be shown that these new chasing algorithms share the computational complexity of their well-known instances the Hessenberg and Hessenberg-like case.

Numerical experiments are conducted, to investigate the unitary similarity transformation to a compressed formaat and to test the *QR*-like method for computing the eigenvalues of the matrices considered.

The article is organized as follows. Section 2 presents preliminary results, definitions and assumptions related to all the essentials for understanding the article. Section 3 discusses the algorithm for carrying out the similarity transformation to a prescribed matrix structure. In Section 4 the new *QR*-like algorithm is presented as an explicit calculation, some issues such as the structure preservation and irreducibility are considered. Section 5 presents an implicit version of the generalized *QR*-method. Some comments on the software and numerical issues and an example are given in Section 6. Conclusions and future work close the article.

2. Preliminary results. To obtain a self-contained article some definitions and techniques for factoring unitary matrices in rotations will be reviewed.

A *Hessenberg* matrix H has all elements below the subdiagonal zero. A *Hessenberg-like* matrix Z has all submatrices taken out of the lower triangular part of rank at most one: $\text{rank}\,(Z(i:n,1:i)) \leq 1$, for all $i = 1,\ldots,n$. In case of invertibility, the inverse of a Hessenberg matrix is of Hessenberg-like form. It is assumed that the reader is familiar with a Givens rotation and its ability of creating zeros in prescribed matrix positions [18] by altering only two rows or columns of the involved matrix.

REMARK 2.1. *Throughout the manuscript we will continually use* 2×2 *rotations, though all results remain valid when using* 2×2 *unitary matrices instead. A rotation has determinant equal to* 1*, whereas the determinant of a unitary matrix can reach an arbitrary value of absolute value equal to* 1.

Rotations will be the building blocks for dealing with factorizations of unitary matrices. To be able to benefit from this rotational factorization it is crucial that an elegant, intuitive and compact manner is provided for keeping track of all information accompagnying an individual rotation. A graphical depiction presents us readily the order of the rotations and indicates unambiguous the rows affected by the rotation.

This graphical representation is introduced by computing the *QR*-factorization of a matrix $A = (a_{ij})_{ij} \in \mathbb{C}^{5\times 5}$. The matrix elements are depicted by the symbol $\times$, each rotation is represented by a square bracket with arrows. The arrows point towards the rows affected when applying the rotation to the matrix. To compute the *QR*-factorization by means of rotations, first the lower left element a_{51} is annihilated by G_{51}^H. The matrix $G_{51}^H A$ has a zero element in position $(5,1)$. Graphically we get:

$$G_{51}^H A = \begin{array}{c} \\ \curvearrowright \\ \\ \\ \\ \end{array} \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix} = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \end{bmatrix}.$$

The procedure is continued by eliminating all, but the top element of the first column. This results in

$G_{21}^H G_{31}^H G_{41}^H G_{51}^H A = Q_1^H A$, graphically depicted as follows:

$$Q_1^H A = \quad \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix} \quad = \quad \begin{bmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \end{bmatrix}.$$

In turn all columns are brought to upper triangular form by a succession of rotations $Q_4^H Q_3^H Q_2^H Q_1^H A = Q^H A = R$.

$$Q^H A = \quad \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix} \quad = \quad \begin{bmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & 0 & \times \end{bmatrix}. \tag{2.1}$$

Premultiplication by Q leads to the QR-factorization of A, graphically represented in condensed format as follows (thereby not depicting zero elements anymore):

$$A = \quad \begin{bmatrix} \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \\ & & & & \times \end{bmatrix}. \tag{2.2}$$

This schematic representation contains all essential information needed for operating with the QR-factorization of the matrix A. The clearly denote on which rows the rotations act and also the mutual order of the rotations is available. Moreover, one can read from these schemes which rotations (e.g. the two rotations in the third, fourth or fifth column of the factorization of Q) commute and can be applied on R simultaneously or in either order. With a *rotational factorization* of Q we refer to a factorization of Q in rotations.

REMARK 2.2. *We note that possibly in Equation 2.2 some rotations need not be performed and are therefore equal to the identity matrix. In the graphical schemes of rotational factorizations, rotations equal to the identity are, however, allowed. Only, when based on prespecified knowledge, such as the matrix structure, the rotation equals the identity, the rotation will not be depicted in the schemes and as such reflect this particular structure.*

Based on the matrix structures, a Hessenberg matrix H and a Hessenberg-like matrix Z admit QR-factorizations of the following form:

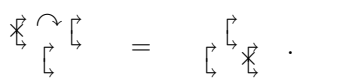
$$H = \quad \begin{bmatrix} \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \\ & & & & \times \end{bmatrix} \quad \text{and} \quad Z = \quad \begin{bmatrix} \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \\ & & & & \times \end{bmatrix}. \tag{2.3}$$

The sequence of rotations in the factorization of H is refered to as a *descending* sequence of rotations. The Hessenberg-like matrix on the contrary admits a rotational factorization of the involved unitary matrix, with an *ascending* sequence of rotations. As will be shown in Section 2.1, a rotational factorization is not always unique. When referring, however, to a very specific ordering of the rotations, as for a descending or ascending sequence, we will speak about the *shape* or the *pattern* of the rotations.

2.1. Manipulations with rotations. The forthcoming algorithms depend heavily on manipulating rotations. These operations are already described and proved elsewhere (see e.g. [32, 33]). For completeness, however, we will briefly reconsider the necessary ones in this section, without proofs.

LEMMA 2.3. *Suppose two 2×2 rotations G_1 and G_2 are given. Their product $G_1 G_2 = G_3$ is again a rotation. This operation is named the fusion of rotations.* Graphically we depict this operation by a curly arrow as follows:

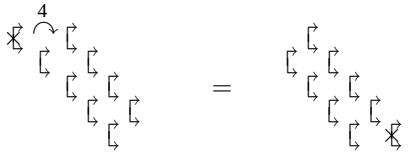
LEMMA 2.4 (Shift-through operation). *Suppose three* 3×3 *rotations* $\check{G}_1,\check{G}_2$ *and* $\check{G}_3$ *are given, such that the rotations* $\check{G}_1$ *and* $\check{G}_3$ *act on the first two rows of a matrix, and* $\check{G}_2$ *acts on the second and third row (when applied on the left to a matrix). Then there exist three rotations* $\hat{G}_1,\hat{G}_2$ *and* $\hat{G}_3$ *satisfying the equality* $\check{G}_1\check{G}_2\check{G}_3=\hat{G}_1\hat{G}_2\hat{G}_3$*, where* $\hat{G}_1$ *and* $\hat{G}_3$ *work on the second and third row and* $\hat{G}_2$ *operates on the first two rows.* This is a common result, whose proof is based on two variants for factoring a 3×3 unitary matrix (see [34]). Schematically this operation is depicted with arrows like $\curvearrowright$ indicating were the marked rotation will go to:



The last important operation is the shift-through operation of length ℓ.

LEMMA 2.5 (Shift-through operation of length ℓ). *Suppose the following matrix product* $\check{G}\check{W}\check{X}$ *is given, where* $\check{G}$ *denotes a rotation acting on row* 1 *and* 2*. The matrices* $\check{W}$ *and* $\check{X}$ *are unitary matrices, both having a rotational factorization in a descending sequence of* ℓ *rotations. The ith rotation* $\check{G}_i^W$ *of* $\check{W}$ *acts on row* $i+1$ *and* $i+2$*. The ith rotation* $\check{G}_i^X$ *related to* $\check{X}$ *acts on row i and* $i+1$*. The matrix product* $\check{G}\check{W}\check{X}$ *can be refactored as* $\check{G}\check{W}\check{X}=\hat{W}\hat{X}\hat{G}$*, where* $\hat{G}$ *is a rotation acting on row* $\ell+1$ *and* $\ell+2$*. The unitary matrices* $\hat{W}$ *and* $\hat{X}$ *are again descending sequences of* ℓ *rotations.*

The next two schemes depict the same unitary matrix, but a different rotational factorization of this matrix. In Scheme (2.4) the rotations $\check{G}$ and $\hat{G}$ are marked with a $\times$. On the left, the matrix $\check{W}$ equals the lower descending sequence of rotations, $\check{X}$ is the upper sequence of rotations. A shift-through operation of a specified length is depicted by adding a superscript to an arrow $\curvearrowright$, indicating the number of successive shift-through operations that need to be performed. The arrow points to the final position of the marked rotation.



(2.4)

On the right of (2.4) we see the resulting scheme in which the rotation $\hat{G}$ appears now on the bottom right, again $\hat{W}$ is the bottom and $\hat{X}$ the upper descending sequence of rotations.

Dealing with *QR*-factorizations requires also to characterize the interaction between rotations and upper triangular matrices *R*. For simplicity we will confine ourselves in this article to nonsingular matrices[1] *A*. The next lemma specifies how sequences of rotations change when transfering them through an upper triangular matrix.

LEMMA 2.6. *A nonsingular* $n\times n$ *complex matrix A, with RQ-factorization* $A=\hat{R}\hat{Q}$ *admits a QR-factorization* $A=QR$*, where the rotational factorizations of Q and* $\hat{Q}$ *obey the same pattern.* A proof of this Lemma can be found in [27]. In words this means that transfering rotations through an upper triangular matrix affects the rotations themselves, but not the mutual position nor the rows/columns they act on.

3. Unitary similarity rotations. Starting from the *QR*-factorization of a matrix $A=QR$, a unitary similarity transformation will be devised such that the unitary factor $\hat{Q}$ in the *QR*-factorization of the resulting matrix $\hat{A}=\hat{Q}\hat{R}$ admits a factorization in $n-1$ rotations appearing in a prescribed order. Admissible outcomes of the similarity transformation, unicity as well as some usefull decompositions will be considered. In the remainder of the text when speaking about a *similarity transformation*, a unitary similarity transformation is considered.

3.1. Admissable outcomes and compressed unitary matrices. Some particulars and extra notation are needed to specify without ambiguity the patterns of factorizations of the unitary matrix *Q* admitted as outcomes of the similarity transformation.

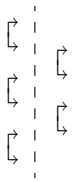
[1]We note that Lemma 2.6 remains valid, also in the singular case [27]. But to exclude some technical and troublesome cases in the forthcoming parts on unitary similarity transformations and the *QR*-like method, this restriction is posed in this article.

It is well-known that any unitary matrix Q is similar to unitary Hessenberg matrix $\hat{Q}$, admitting a Schur parametrization [9]. The Schur parametrization is a multiplicative factorization of $\hat{Q}$ into a descending sequence of $n-1$ rotations. The forthcoming definition of a compressed unitary matrix is sort of extension of this factorization.

A vector $\mathbf{p}$ of length $n-2$ containing the characters ℓ (left) and r (right) will define the strict order of the rotations. Suppose $n-1$ rotations $G_i, i = 1,\ldots,n-1$ are given where G_i operates on rows i and $i+1$. The ith element $\mathbf{p}_i$ of the *position vector* $\mathbf{p}$ specifies the position of the factor G_i w.r.t. the position of the factor G_{i+1} in the product of these rotations.

For example, the unitary matrix Q in the QR-factorization of a Hessenberg matrix H is factored in a descending sequence of rotations. This means that G_i always appears on the left of G_{i+1}, hence $Q = G_1G_2\ldots G_{n-1}$ and the corresponding $\mathbf{p} = [\ell,\ldots,\ell]$. A Hessenberg-like matrix, however, is related to an ascending sequence of iterations. Therefore $Q = G_{n-1}\ldots G_2G_1$ and $\mathbf{p} = [r,\ldots,r]$. The CMV-decomposition [10, 23] of a pentadiagonal unitary matrix $P = UV$ factors the matrix in two block diagonal matrices U and V, where the blocks are 2×2 rotations.

$$P = UV = \tag{3.1}$$



Corresponding to our notation[2] $P = G_5G_3G_1G_2G_4$, where $U = G_5G_3G_1$ and $V = G_2G_4$. The corresponding vector $\mathbf{p}$ equals $[\ell,r,\ell,r]$.

In the rest of the article we refer to a *compressed unitary* matrix as a unitary matrix admitting a factorization of $n-1$ rotations G_i acting on rows i and $i+1$, according to a prescribed ordering $\mathbf{p}$. The forthcoming similarity transformation admits any type of compressed unitary matrix $\hat{Q}$ in the QR-factorization of the resulting matrix $\hat{A} = \hat{Q}\hat{R}$, covering thereby the Hessenberg, Hessenberg-like and the CMV case as particular examples.

Factorizations of a unitary matrix Q in rotations were discussed before in several articles [1, 2, 4, 9, 10, 37]. The origin lies in the so-called schur parametrization [9]. Already in these manuscripts results related to other types of patterns [1] were proposed. The relation between orthogonal Laurent polynomials and the corresponding unitary matrices were studied in [4, 10, 37]. Depending on the order of the monomials for constructing an orthogonal basis of Laurent polynomials the recurrences are given by a compressed unitary matrix. Both the order of the monomials as well as the shape of the corresponing compressed unitary matrix are determined by a position vector $\mathbf{p}$. In [4], these factorizations are refered to as snake shaped matrix factorizations due to the zigzag form of the pattern of rotations.

REMARK 3.1. *Though not explicitly mentioned, it is tacitly assumed that all unitary matrices have determinant equal to* 1. *Otherwise an extra unimodular factor appears when building the factorization of a compressed unitary matrix. This unimodular factor does not pose any difficulties, all forthcoming deductions remain valid just as the shift-through operations and the fusions, only all theorems need the incorporation of a unimodular factor. Instead of considering rotations one can also use* 2×2 *unitary matrices solving the problem immediately.*

3.2. The algorithm. One can consider the piramid shaped factorization of a generic unitary matrix Q in (2.2) as made up by 4 descending sequences (2.1) or as made up by 4 ascending sequences of rotations. This two-folded interpretation enables us to reduce any matrix to the desired compressed format specified by $\mathbf{p}$. First the algorithm is presented, followed by a justification.

In each step i (for $i = 1,\ldots,n-1$) of the algorithm the following similarity transformation is executed[3]:

- (if $\mathbf{p}_i = r$) annihilate the outer $n-1-i$ rotations of the rightmost yet untreated sequence of descending rotations by a similarity transformation;

[2] Another interpretation could lead to a factorization $P = G_1G_3G_5G_4G_2$, which is, however, due to commutativity equivalent to $P = G_5G_3G_1G_2G_4$.

[3] Of course, in case a matrix is already structured, some transformations need not be executed. Assume therefore a generic full piramid shaped factorization of the matrix Q.

- (if $\mathbf{p}_i = \ell$) annihilate the outer $n-1-i$ rotations of the leftmost yet untreated sequence of ascending rotations by a similarity transformation.

The outcome of these $n-1$ steps result in a similar matrix $\hat{A} = \hat{Q}\hat{R}$ where the factorization of $\hat{Q}$ in rotations obeys the pattern identified by $\mathbf{p}$.

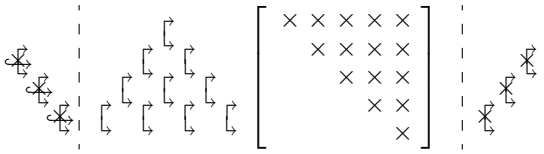
There are two building blocks in the design of this algorithm, annihilation of rotations on the left or either on the right. First, in Sections 3.2.1 and 3.2.2 a single left and right annihilation step is considered on a piramid pattern coming from the *QR*-factorization of a 5×5 matrix. The combination of these steps is presented in Section 3.2.3.

3.2.1. Left annihilation of a sequence. Consider $A = A^{(0)} = Q^{(0)}R^{(0)}$, factored as in (2.2). Assume $\mathbf{p}_1 = \ell$, requiring the annihilation of the 3 leftmost iterations. The 3 leftmost rotations $G_1^{(0)}G_2^{(0)}G_3^{(0)}$ in the piramid shaped factorization of $Q^{(0)}$ define the similarity transformation

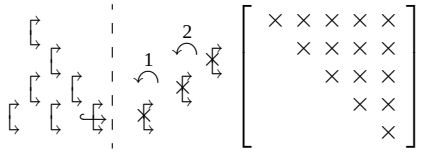
$$A^{(1)} = G_3^{(0)H}G_2^{(0)H}G_1^{(0)H}\left(Q^{(0)}R^{(0)}\right)G_1^{(0)}G_2^{(0)}G_3^{(0)} = Q^{(1)}R^{(1)}, \tag{3.2}$$

where $Q^{(1)}R^{(1)}$ is the *QR*-factorization of $A^{(1)}$. Note that $Q^{(1)}$ must admit a rotational factorization without the three leftmost rotations and no extra rotations on the right remaining. The forthcoming procedure will update the factorization of the matrix $Q^{(0)}$ to obtain $Q^{(1)}$.

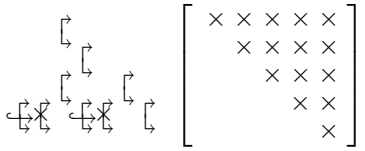
The new *QR*-factorization of $A^{(1)}$ is computed as follows. The rotations determining the similarity transformation are marked by $\times$ and separated from the *QR*-factorization by a vertical dashed line. In (3.3) Formula (3.2) is depicted.

$A^{(1)} =$  (3.3)

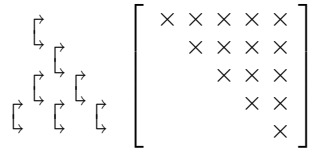
Since the marked rotations are constructed such to annihilate three rotations in the left of the piramid shape, six rotations in total will vanish. The rotations on the right are transfered through the upper triangular matrix $R^{(0)}$ by using Lemma 2.6.

$A^{(1)} =$  (3.4)

In Scheme (3.4) three more undesired rotations (marked by $\times$) are present. The bottom rotation can be incorporated in the rotation preceding it by a fusion. The remaining two rotations are moved downwards by respectively a single and a double shift-through operation. The result is a *QR*-factorization of the following form.

$A^{(1)} =$  (3.5)

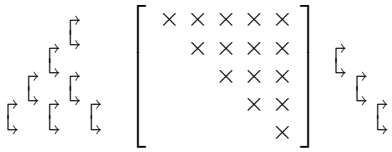
Two more fusions in Scheme (3.5) are required to obtain the desired rotational factorization of the matrix $Q^{(1)}$ which has now 3 rotations less than $Q^{(0)}$. The *QR*-factorization of $A^{(1)} = Q^{(1)}R^{(1)}$ is of the form

$A^{(1)} =$  .

It is important to observe that the top rotation of the piramid remains unaltered (it does not even change) during the whole process. We will see that also in the right annihilation process the top rotation remains untouched.

3.2.2. Right annihilation of a sequence. Take $A^{(0)} = Q^{(0)}R^{(0)}$ again as in (2.2). Now $\mathbf{p}_1 = r$, implying the removal of 3 rotations on the right. Whereas in the left annihilation step the rotations determining the similarity transformation are readily available, this is not the case here. A sort of preliminary step needs to be performed, transfering the rotations prone to removal to the right of the upper triangular matrix. By Lemma 2.6, Scheme (2.2) is refactored and thereby transformed in $A^{(0)} = Q^{(0)}R^{(0)} = \tilde{Q}^{(0)}\tilde{R}^{(0)}G_3^{(0)H}G_2^{(0)H}G_1^{(0)H}$.

$$A^{(0)} = \qquad (3.6)$$



The three rotations on the right $G_3^{(0)H}G_2^{(0)H}G_1^{(0)H}$ determine the forthcoming similarity transformation:

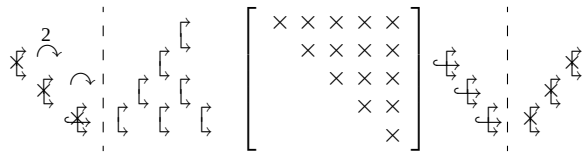
$$A^{(1)} = G_3^{(0)H}G_2^{(0)H}G_1^{(0)H}\left(Q^{(0)}R^{(0)}\right)G_1^{(0)}G_2^{(0)}G_3^{(0)} \qquad (3.7)$$

$$= G_3^{(0)H}G_2^{(0)H}G_1^{(0)H}\left(\tilde{Q}^{(0)}\tilde{R}^{(0)}G_3^{(0)H}G_2^{(0)H}G_1^{(0)H}\right)G_1^{(0)}G_2^{(0)}G_3^{(0)} \qquad (3.8)$$

$$= G_3^{(0)H}G_2^{(0)H}G_1^{(0)H}\left(\tilde{Q}^{(0)}\tilde{R}^{(0)}\right) = Q^{(1)}R^{(1)},$$

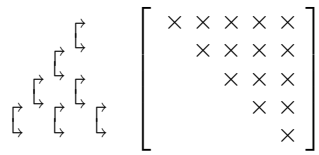
where $Q^{(1)}$ must admits now a rotational factorization without the three rightmost rotations. The forthcoming operations are quite similar to the ones for a left annihilation step and provide us the new rotational QR-factorization of $A^{(1)} = Q^{(1)}R^{(1)}$. After the similarity transformation the following equation is obtained corresponding to factorization (3.8):

$$A^{(1)} = \qquad$$



The six rightmost rotations vanish after the fusions. The undesired rotations on the left are dealt with similarly as in Section 3.2.1: after a fusion, some shift-through operations and two more fusions they vanish and as a result the new factorization of $A^{(1)} = Q^{(1)}R^{(1)}$ is retrieved and the matrix $Q^{(1)}$ satisfies the constraints.

$$A^{(1)} = \qquad$$

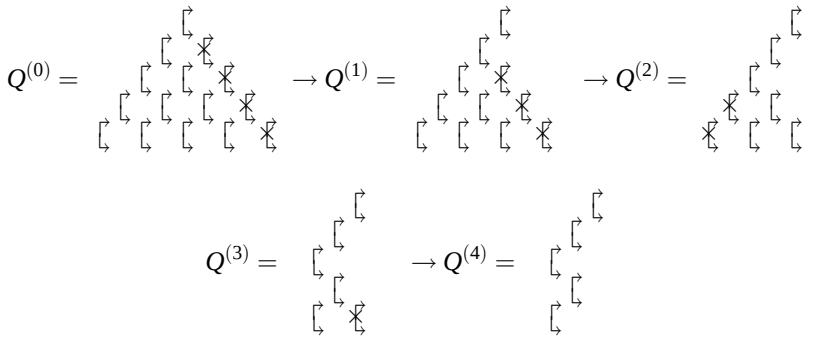


Again we note that the rotation on top of the piramed is not altered by any of the performed operations.

3.2.3. Generic algorithm. The generic algorithm is illustrated by a simple example. The clue is that after each annihilation step the top rotation stays in place. Not considering this top rotation leaves us with a new piramid of one dimension less on which we can again apply one of the two annihilation steps described above. Once more the top of the smaller piramid remains and the procedure continues.

A formal proof that the algorithm produces the exact desired output is done by induction. For a left annihilation in step i, the smaller piramid is located on the right of the remaining top rotation, hence the rotation G_i is positioned on the left of the top rotation G_{i+1} of the smaller piramid. Note that the position is fixed now, since G_{i+1} will stay in place. For a right annihilation obviously G_i will be positioned on the right of the next remaining rotation G_{i+1}. Hence, after $n-1$ steps, the order of the rotations is fixed and the requested outcome is obtained.

Let us consider for example a 6×6 matrix and a reduction process specified by the vector $\mathbf{p}=[r,r,l,r]$. For simplicity, only the rotational factorization of the unitary matrix $Q^{(i)}$ is depicted in each step. In each step i the rotations prone to removal (specified by $\mathbf{p}_i$) are marked by a $\times$.



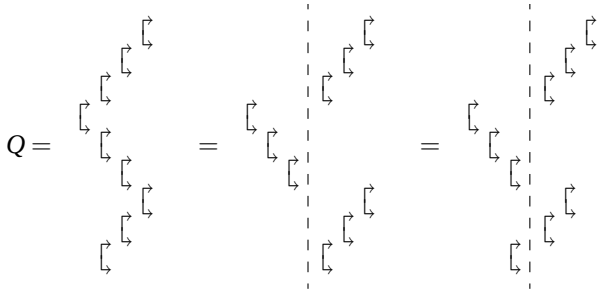
As a result we obtain a QR-factorization $A^{(4)}=Q^{(4)}R^{(4)}$, where $Q^{(4)}$ is factored by $n-1=5$ rotations satisfying the desired pattern.

3.3. The zigzag pattern decomposed in ascending-descending format. Section 4 presents an alternative type of QR-algorithm suitable for any type of matrix $A=QR$ with Q in compressed format. To make the description of the algorithm comprehensible and compact, two factorizations based on the zigzag pattern will be introduced.

A compressed unitary matrix Q admits a rotational factorization decomposed in two factors Q_d and Q_a: $Q=Q_dQ_a$, where Q_a consists of a single sequence of ascending rotations and Q_d consists of a single sequence of descending rotations.

The mutual position between two successive rotations G_i and G_{i+1} regulates the factorization. The unitary matrix Q_d contains the rotation G_i if rotation G_i precedes G_{i+1} (G_i is located on the left of G_{i+1}, hence $\mathbf{p}_i=\ell$), otherwise G_i is stored in Q_a. Since the trailing rotation G_{n-1} has no successor, it can be put in either unitary matrix.

EXAMPLE 3.2. *Consider the unitary matrix Q in* (3.9) *with the given zigzag pattern of rotations. The two possible factorizations, depending on the position of the trailing rotation, of Q are shown. The rotations belonging to Q_d are on the left of the dashed line, the rotations of Q_a on the right. Based on commutation properties the rotations can be reordered a little such that visually a single descending and a single ascending sequence of rotations is obtained.*



(3.9)

Another example of a *descending-ascending (DA-)factorization* is the *CMV*-decomposition (see (3.1)).

REMARK 3.3. *Any compressed unitary matrix admits also an ascending-descending (AD-)factorization $Q=Q_aQ_d$. The unitary matrix Q_a contains the rotation G_i if rotation G_i precedes G_{i-1} (G_i is positioned on the left of G_{i-1}). Otherwise G_i is stored in Q_d. Since the first rotation G_1 has no precessor, it can be put in either unitary matrix.*

REMARK 3.4. *The forthcoming QR-like algorithm is based on the DA-factorization, but also the AD-factorization can be used without any essential changes taking place. For example, the chasing technique (see Section 5) for a DA-factorization goes from to to bottom and for a DA-factorization it runs from bottom to top. Up to a certain sense the links are similar to the ones between GR-like and RG-like iterations.*

To avoid the cumbersome and even misleading wording "*QR*-like method", we refer to the forthcoming iterative algorithm for computing the eigenvalues as the *DA-algorithm*.

3.4. Unicity of the similarity transformation. An important, yet unanswered question is the one of unicity. The theorem formulated here is sort of extension of the implicit *Q*-theorem, since also the Hessenberg case is covered. The proof provided here is based on Krylov matrices as in [40], which is more appealing than direct calculations as in [18, 32] for Hessenberg and Hessenberg-like matrices. As mentioned before, for simplicity, we restrict ourselves in this article to nonsingular matrices.

THEOREM 3.5 (Implicit *Q*-theorem). *Consider a nonsingular matrix A and a position vector* $\mathbf{p}$. *Let V_1 and V_2 be two unitary matrices sharing the same first column (up to a unimodular factor) such that*

$$Q_1R_1 = A_1 = V_1^H A V_1 \quad \text{and} \quad Q_2R_2 = A_2 = V_2^H A V_2,$$

where the unitary factors Q_1 and Q_2 in the QR-decompositions of A_1 and A_2 obey the pattern specified by $\mathbf{p}$ *and all rotations appearing in these factorizations of Q_1 and Q_2 are different from the identity*[4]. *Then we have that both matrices A_1 and A_2 are essentially identical.* The zigzag patterns arising in the various factorizations of compressed unitary matrices have a close relation to the ordering of monomials when considering the recurrence relations for orthogonal Laurent polynomials [4]. Here the ordering of multiplication with A or A^{-1} will play an important role in the construction of the involved rational Krylov matrices.

We consider rational Krylov sequences $\mathscr{K}_{\mathbf{p},k}$ of length k, with starting vector $\mathbf{v}$ spanned by k vectors out of the sequence

$$\ldots,\ A^3\mathbf{v},\ A^2\mathbf{v},\ A\mathbf{v},\ \mathbf{v},\ A^{-1}\mathbf{v},\ A^{-2}\mathbf{v}, A^{-3}\mathbf{v},\ \ldots \tag{3.10}$$

the order in which the vectors appear in the sequence is determined by the position vector $\mathbf{p}$. The first vector in the Krylov sequence is always the middle vector $\mathbf{v}$. If $\mathbf{p}_i = \ell$ the next vector on the left is taken to fill up position $i+1$, if $\mathbf{p}_i = r$, one takes the next one on the right. Since $\mathbf{p}_{n-1}$ is not specified, the last vector, the one in the nth position can either be from the left or from the right, this is an optional choice.[5] The associated Krylov matrix $K_{\mathbf{p},k}(A,\mathbf{v})$, has these vectors generating the sequence as columns of a $n \times k$ matrix. When $k = n$, k is omitted as subscript in $\mathscr{K}$ and K

For example, the *CMV*-pattern in (3.1) is determined by $\mathbf{p} = [\ell, r, \ell, r]$. The corresponding rational Krylov sequence is $\mathscr{K}_{\mathbf{p},5}(A,\mathbf{v}) = \text{span}\{\mathbf{v},\ A\mathbf{v},\ A^{-1}\mathbf{v},\ A^2\mathbf{v},\ A^{-2}\mathbf{v}\}$.

The proof of Theorem 3.5 is decomposed in two parts. Consider a matrix A reduced via the similarity transformation from Section 3 to a matrix factorization $V^H A V = \hat{A} = \hat{Q}\hat{R}$ according to the position vector $\mathbf{p}$. Crucial in the proof of Theorem 3.5 is the observation that $K_{\mathbf{p}}(\hat{A}, \mathbf{e}_1)$ is upper triangular, which forms the first and most technical part of the proof. Once this is known, the second part proceeds identical as in [40].

3.4.1. Upper triangular Krylov matrix. Crucial is that all rotations in the rotational factorization of the compressed unitary matrix $\hat{Q}$ are different from zero. Otherwise the proof will break-down and essential uniqueness can only be proved up to a certain point. To prove that $K_{\mathbf{p}}(\hat{A}, \mathbf{e}_1)$ is upper triangular the *DA*-factorization of $\hat{Q} = \hat{Q}_d\hat{Q}_a$ is utilized. Both $\hat{Q}_d$ and $\hat{Q}_a$ are build up by nonoverlapping diagonal blocks, in which each block contains a sequences of rotations (see (3.9)), the location of these blocks is vital for the proof.

We will investigate the zero-structure of the vectors $\hat{A}^k\mathbf{e}_1$, using the *DA*-decomposition of $\hat{Q}$. The analysis of $\hat{A}^{-k}\mathbf{e}_1$ proceeds similarly. Since $\hat{Q}^{-1} = \hat{Q}_a^{-1}\hat{Q}_d^{-1}$ is a *DA*-decomposition of $\hat{Q}^{-1}$ and by Lemma 2.6 it is known that the *DA*-decomposition of the *Q*-factor of the *QR*-factorization of $\hat{A}^{-1}$ has the same pattern as the *DA*-decomposition $\hat{Q}_a^{-1}\hat{Q}_d^{-1}$. This provides us all the essentials for analyzing the structure of the vectors $\hat{A}^{-k}\mathbf{e}_1$ in essentially the same way as for $A^k\mathbf{e}_1$.

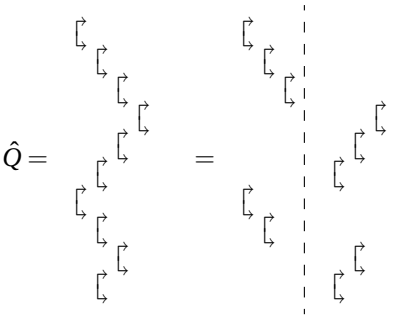
As before, the vectors $\mathbf{e}_i$ denote the standard basis vectors; the vectors $\tilde{\mathbf{e}}_i$ stand for vectors having the element in position i different from zero, the elements below i are zero and the elements above might be

[4]This is related to irreducibility and we will come back to this in Section 4.1. When an identity rotation is considered unicity is only guaranteed up to this position, just like in the standard implicit *Q*-theorem.

[5]We point out that the freedom of the trailing items is recurring throughout the article.

nonzero. In fact $\tilde{\mathbf{e}}_i$ equals a linear combination of the $\mathbf{e}_j$, $1 \leq j \leq i$, where the coefficient of $\mathbf{e}_i$ is nonzero, the other coefficients are unspecified.

The study of the structure of $K_{\mathbf{p}}(\hat{A}, \mathbf{e}_1)$ is quite technical, therefore the flow of the proof will be accompagnied by an example in which $\mathbf{p} = [\ell, \ell, \ell, r, r, r, \ell, \ell, r]$. A *DA*-decomposition of $\hat{Q}$ corresponding to the example is of the form.

$\hat{Q} =$  (3.11)

The indices i_j are defined as follows: $i_1 = 1$ and i_j is the position k such that the jth transition from ℓ to r or r to ℓ takes place between $\mathbf{p}_{k-1}$ and $\mathbf{p}_k$. The list is closed by a trailing $i_j = n$, where n is the dimension of $\hat{Q}$. In the example we have $i_1 = 1, i_2 = 4, i_3 = 7, i_4 = 9$ and $i_5 = 11$. For simplicity we assume in the forthcoming part that $\mathbf{p}_1 = \ell$, if it equals r, only a change between even and odd in the upcoming part is required. Based on this indexing, we can identify diagonal blocks in the matrices $\hat{Q}_a$ and $\hat{Q}_d$: the submatrices $\hat{Q}_d(i_j : i_{j+1})$ (for odd j) are of Hessenberg form; the submatrices $\hat{Q}_a(i_j : i_{j+1})$ (for even j) are of Hessenberg-like form; the remaining diagonal blocks in both $\hat{Q}_a$ and $\hat{Q}_d$ equal the identity. Please note that for both matrices $\hat{Q}_d$ and $\hat{Q}_a$ the elements in the positions (i_j, i_j) are included in a Hessenberg (for $\hat{Q}_d$) and a Hessenberg-like structure (for $\hat{Q}_a$).

The derivations in the forthcoming part are only valid when all rotations are nonidentical. Based on the *DA*-factorization and the indexset, one can verify that (thereby using several times Lemma 2.6)

$$\hat{A}^k \mathbf{e}_1 = \tilde{\mathbf{e}}_{k+1} \quad \text{for } i_1 \leq k \leq i_2 - 1,$$

since the entire block $\hat{Q}_a(i_1 : i_2 - 1)$ equals the identity.

Considering, however, $\hat{A}^{i_2}$, the matrix $\hat{Q}_a$ will come into play since it has a Hessenberg-like block on the diagonal, ranging from position i_2 to i_3. This interaction with $\hat{Q}_a$ will smear out nonzero elements immediately untill position i_3. This means that we get that (there is a little abuse in notation by considering $\hat{R}\tilde{\mathbf{e}}_{i_2} = \tilde{\mathbf{e}}_{i_2}$, but in the spirit of the undefinedness of some elements in the vector $\tilde{\mathbf{e}}_{i_2}$ this is appropriate)

$$\hat{A}^{i_2} \mathbf{e}_1 = \hat{A}\, \tilde{\mathbf{e}}_{i_2} = \hat{Q}_d\, \hat{Q}_a\, \hat{R}\, \tilde{\mathbf{e}}_{i_2} = \hat{Q}_d\, \hat{Q}_a\, \tilde{\mathbf{e}}_{i_2} = \hat{Q}_d \tilde{\mathbf{e}}_{i_3} = \tilde{\mathbf{e}}_{i_3+1}.$$

The action of $\hat{Q}_d$ on $\tilde{\mathbf{e}}_{i_3}$ is properly defined due to the Hessenberg block $\hat{Q}_d(i_3 : i_4)$. Roughly speaking one can say that diagonal block in the matrix $\hat{Q}_a$ shifts down the nonzero elements quite dramatically untill position i_3. From this position there is temporarly again no impact of $\hat{Q}_a$, meaning that the Hessenberg structure of $\hat{Q}_d$ can again shift down the nonzero elements in the vector one by one. The Hessenberg part in action is now $\hat{Q}_d(i_3 : i_4)$. Unfortunately, once position i_4 is reached $\hat{Q}_a$ comes in action again. Combining all this information the following formulas are retrieved:

$$\hat{A}^{k-i_3+i_2} \mathbf{e}_1 = \tilde{\mathbf{e}}_{k+1} \quad \text{for } i_3 \leq k \leq i_4 - 1,$$

indicating that the powers from $\hat{A}$ continue to grow from i_2, but the nonzero elements have shifted down until the next i_j for j odd.

Putting all things together the following formula predicts the nonzero structure of the powers of $\hat{A}$ applied on $\mathbf{e}_1$ (take $i_0 = 1$):

$$\hat{A}^{k+\Sigma_{\hat{j}\, odd\ \&\ \hat{j}\leq j}(i_{\hat{j}-1} - i_{\hat{j}})}\ \mathbf{e}_1 = \tilde{\mathbf{e}}_{k+1} \quad \text{for } i_j \leq k \leq i_{j+1} - 1, \text{ for } j \text{ odd}.$$

In words this simply states that when an i_j is reached for even j, that there is a downward shift of the nonzeros up to position i_j. The somehow complicated notation of the power just expresses that the power

keeps increasing by one each step. For our particular example the following relations are obtained:

$$\hat{A}\mathbf{e}_1 = \tilde{\mathbf{e}}_2,\ \hat{A}^2\mathbf{e}_1 = \tilde{\mathbf{e}}_3,\ \hat{A}^3\mathbf{e}_1 = \tilde{\mathbf{e}}_4,\ \hat{A}^4\mathbf{e}_1 = \tilde{\mathbf{e}}_8,\ \hat{A}^5\mathbf{e}_1 = \tilde{\mathbf{e}}_9,\ \hat{A}^6\mathbf{e}_1 = \tilde{\mathbf{e}}_{11},$$

For the inverse powers $\hat{A}^{-k}$ similar formulas are obtained:

$$\hat{A}^{-(k+\Sigma_{\hat{j}\,even\ \&\ \hat{j}\le j}(i_{\hat{j}-1}-i_{\hat{j}}))}\ \mathbf{e}_1 = \tilde{\mathbf{e}}_{k+1} \quad \text{for } i_j \le k \le i_{j+1}-1, \text{ for } j \text{ even}.$$

The example gives in this case:

$$\hat{A}^{-1}\,\mathbf{e}_1 = \tilde{\mathbf{e}}_5,\ \hat{A}^{-2}\,\mathbf{e}_1 = \tilde{\mathbf{e}}_6,\ \hat{A}^{-3}\,\mathbf{e}_1 = \tilde{\mathbf{e}}_7,\ \hat{A}^{-4}\,\mathbf{e}_1 = \tilde{\mathbf{e}}_{10},\ \hat{A}^{-5}\,\mathbf{e}_1 = \tilde{\mathbf{e}}_{11},$$

Glueing now all the results for the powers and inverse powers together, taking thereby the ordering of the position vector $\mathbf{p}$ into account, one gets:

$$K_{\mathbf{p},k}(\hat{A}) = [\tilde{\mathbf{e}}_1, \ldots, \tilde{\mathbf{e}}_k].$$

Indeed testing it on our example we get:

$$\begin{aligned} K_{\mathbf{p}}(\hat{A}, \mathbf{e}_1) &= [\mathbf{e}_1, A\mathbf{e}_1, A^2\mathbf{e}_1, A^3\mathbf{e}_1, A^{-1}\mathbf{e}_1, A^{-2}\mathbf{e}_1, A^{-3}\mathbf{e}_1, A^4\mathbf{e}_1, A^5\mathbf{e}_1, A^{-4}\mathbf{e}_1, A^{-5}\mathbf{e}_1] \\ &= [\mathbf{e}_1, A\mathbf{e}_1, A^2\mathbf{e}_1, A^3\mathbf{e}_1, A^{-1}\mathbf{e}_1, A^{-2}\mathbf{e}_1, A^{-3}\mathbf{e}_1, A^4\mathbf{e}_1, A^5\mathbf{e}_1, A^{-4}\mathbf{e}_1, A^6\mathbf{e}_1] \\ &= [\tilde{\mathbf{e}}_1, \ldots, \tilde{\mathbf{e}}_{11}]. \end{aligned}$$

This concludes the technical part for proving that $K_{\mathbf{p}}(\hat{A}, \mathbf{e}_1)$ is upper triangular.

3.4.2. Unicity based on the *QR*-factorization. By the definition of the rational Krylov matrices and using $\hat{A} = V^H A V$ we get:

$$V K_{\mathbf{p}}(\hat{A}, \mathbf{e}_1) = K_{\mathbf{p}}(V\hat{A}V^H, V\mathbf{e}_1) = K_{\mathbf{p}}(A, V\mathbf{e}_1). \tag{3.12}$$

Using the notation of Theorem 3.5 the following equalities are obtained (σ is a unimodular factor) :

$$\begin{aligned} V_1 K_{\mathbf{p}}(A_1, \mathbf{e}_1) = K_{\mathbf{p}}(V_1 A_1 V_1^H, V_1\mathbf{e}_1) &= K_{\mathbf{p}}(A, V_1\mathbf{e}_1) \\ &= \sigma K_{\mathbf{p}}(A, V_2\mathbf{e}_1) \\ &= \sigma K_{\mathbf{p}}(V_2 A_2 V_2^H, V_2\mathbf{e}_1) = \sigma V_2 K_{\mathbf{p}}(A_2, \mathbf{e}_1). \end{aligned}$$

The left and the right term are both *QR*-factorizations. The essential uniqueness of this factorization implies that V_1 and V_2 are essentially identical proving thereby Theorem 3.5.

We will not elaborate on this, but the above relations clearly indicate that the corresponding Krylov sequences uniquely determine the similarity transformation to bring A to $\hat{A}$ format, which means that the Q-factor in the *QR*-factorization of the rational Krylov sequence can also be used to unitarily transform the matrix to the desired format.

4. Eigenvalue computations. Given a matrix A and a suitable shift μ, chosen to enhance the convergence. A single shifted *QR*-step is determined by:

$$A - \mu I = QR \tag{4.1}$$

$$\hat{A} = RQ + \mu I, \tag{4.2}$$

where I stands for the identity matrix and $\hat{A}$ is the outcome and new iterate. Equation (4.1) corresponds to the explicit version of the *QR*-algorithm, where the matrices Q and R are computed explicitly before the reverse product is computed. The implicit version determines the unitary matrix Q on the fly and performs the following transition, without explicitly forming the unitary matrix Q

$$\hat{A} = Q^H A Q. \tag{4.3}$$

One can also define the matrix $\hat{A}$ as

$$\hat{A} = RAR^{-1}, \tag{4.4}$$

this formula is, however, merely of a theoretical interest for proving for instance the preservation of the structure of the matrix A.

4.1. Irreducibility. A matrix A with QR-factorization $A = QR$, where the unitary matrix Q is in compressed format is said to be irreducible if and only if the upper triangular matrix has all diagonal elements, except the trailing one, different from zero and if the rotations involved in the factorization of the unitary matrix Q are different from the identity. It is a simple exercise to prove that this definition is equivalent with the standard definition of irreducibility of Hessenberg and Hessenberg-like matrices. A rotation equal to the identity implies that the matrix is block upper triangular and as such can be split in submatrices for computing the eigenvalues.

4.2. The new iteration. Consider the matrix A with a QR-factorization having the unitary matrix in compressed form and DA-factored: $A = Q_d\tilde{Q}_a\tilde{R}_\ell$. The shifted DA-iteration proposed here, consists of the following steps. First the ascending sequence of rotations $\tilde{Q}_a$ is brought to the right of the upper triangular matrix $\tilde{R}_\ell$ by Lemma 2.6: $\tilde{Q}_a\tilde{R}_\ell = R_rQ_a$.

$$A - \mu I = Q_d\tilde{Q}_a\tilde{R}_\ell - \mu I = \left(Q_dR_r - \mu Q_a^H\right)Q_a = QRQ_a. \tag{4.5}$$

Here the product QR represents the QR-factorization of the term between the brackets. This unitary matrix Q determines the similarity transformation of the DA-step: the new iterate is of the form $\hat{A} = Q^HAQ$. In a certain sense this is sort of a URV-factorization of the matrix $A-\mu I$, where U determines the next similarity transformation. We remark that the DA-factorization admits two variants, depending on the position of the final rotation, as a result there are also two variants for executing a DA-step.

Let us elaborate a little on the structure of Q. The number of rotations appearing in the rotational factorization of the unitary matrix Q is essential for keeping the computational complexity under control; here Q admits a factorization in a single descending sequence of rotations, since $Q_dR_r - \mu Q_a^H$ is an irreducible Hessenberg matrix. Descending sequences of rotations applied on an upper triangular matrix result in a Hessenberg matrix. Hence, two Hessenberg matrices are obtained: $H_\ell = Q_dR_r$ and $H_r = \mu Q_a^H$. Based on the diagonal blocks in the matrices Q_a and Q_d, one can partition the matrices H_ℓ and H_r. It is easily verified that the diagonal blocks of upper triangular form in H_ℓ correspond with diagonal blocks of Hessenberg form in H_r and vice versa. Therefore the summation of both Hessenberg matrices leads to an Hessenberg matrix $H = H_\ell + H_r$. In case A is irreducible, the Hessenberg matrix is irreducible as well, admitting thus an essentially unique QR-factorization build up by $n-1$ rotations.

Before proceeding the analysis of this iteration, the formulas equivalent to (4.1), (4.3) and (4.4) are given. Straightforward calculations imply the following relations, using thereby Equation (4.5):

$$\hat{A} = (RQ_a)Q + \mu I, \quad \hat{A} = Q^HAQ, \quad \text{and} \quad \hat{A} = RQ_aAQ_a^HR^{-1}. \tag{4.6}$$

Both the QR-algorithm for Hessenberg matrices and the rational driven QR-iteration for Hessenberg-like matrices fit nicely in this framework. Consider for example a Hessenberg matrix $H = \tilde{Q}\tilde{R}$. Take the following DA-decomposition $Q_d = \tilde{Q}$ and $\tilde{Q}_a = I$, this means that the final rotation is incorporated in the descending sequence. Reconsidering (4.5) for H gives us ($\tilde{R}_\ell = \tilde{R}$):

$$A - \mu I = \tilde{Q}\tilde{R} - \mu I = QR,$$

as a result $\hat{A} = Q^HAQ$, which is a step of the standard QR-algorithm. For a Hessenberg-like matrix $A = \tilde{Q}\tilde{R}$, the matrix $\tilde{Q}$ is an ascending sequence of rotations. The DA-decomposition considered is $\tilde{Q} = I\tilde{Q}_a$, with the final rotation put in the ascending part of the factorization. Rewriting the formulas from (4.5) gives us ($\tilde{R}_\ell = \tilde{R}$):

$$A - \mu I = I\tilde{Q}_a\tilde{R}_\ell - \mu I = \left(R_r - \mu Q_a^H\right)Q_a = QRQ_a.$$

The new iterate $\hat{A}$ corresponds to a step of the QH-algorithm [31, 35] or the rational driven QR-iteration on the matrix A.

4.3. Structure of the rotational factorization pattern after an iteration. One of the crucial features of the QR-iteration applied on Hessenberg and Hessenberg-like matrices is the preservation of the structure. Consider, however, a nonsingular matrix $A = QR$, with Q in compressed format. Based on (4.4) and Lemma (2.6) it is trivial to prove that the patterns in the factorization of the matrix Q are preserved under

QR-steps. For a general treatment of structures preserved under *QR*-iterates we refer to [13], the singular case is treated thoroughly in [12].

The main reason why we did not focus on the development of *QR*-algorithms for these various types of patterns is the number of rotations needed for determining the similarity transformation. Consider for example a Hessenberg-like matrix. Executing a step of the traditional *QR*-method on this matrix involves the computation of $2n-3$ rotations. A single shifted *QR*-step for a Hessenberg matrix (the inverse of a Hessenberg-like) only needs $n-1$ rotations. One can roughly claim that for dealing with an ascending sequence of rotations the double amount of rotations in a *QR*-step are needed. Another reason is that these matrices belong to the more general class of quasiseparable matrices (also Hessenberg and Hessenberg-like matrices can be considered as quasiseparable matrices). For quasiseparable matrices explicit *QR*-algorithms, determined by much more than $n-1$ rotations each step, are readily available [11, 16, 33].

The *DA*-algorithm proposed here is not precisely the same as the *QR*-method, hence we cannot rely on the structure preserving theorems from [13] and [12]. In fact, it will be shown, that generically the structure is not preserved under an iteration, but alters after each of the iterations. We are particularly interested in the shape of the rotational factorization of the matrix $\hat{Q}$ in the *QR*-factorization of the matrix $\hat{A} = \hat{Q}\hat{R}$ as the result of a *DA*-step, since this determines the matrix structure.

Given a matrix A, the result $\hat{A}$ after a step of the *DA*-iteration is of the form: $\hat{A} = RQ_aAQ_a^HR^{-1}$. Using Lemma 2.6, we know that the upper triangular matrices R and R^{-1} have no impact on the pattern of the Q-factor in the *QR*-factorization of the matrix $\hat{A}$. It remains to investigate the structure of $Q_aAQ_a^H$. Using $A = Q_dR_rQ_a$, we get that the Q-factor in the factorization of $\hat{A}$ shares the pattern of Q_aQ_d.

So the original pattern in the rotational *QR*-factorization of A equals the pattern of Q_dQ_a, whereas the new pattern coming from $\hat{A}$ is equivalent to the pattern of Q_aQ_d. It is easily checked that this reverting of the order implies that the pattern moves up one position (consider e.g. (3.9) and an example is given in (5.5)), with the position of the last rotation determined by the variant of the *DA*-factorization taken to determine the *DA*-step. The flexibility of the position of the last rotation in the decomposition of Q_dQ_a implies that one can alter the pattern on the fly, since both choices determine also slightly different resulting structures. This flexibility of the final rotation in the *DA*-factorization appears in the final column of the Krylov matrices and in the position of the final rotation in the structure preservation. Moreover, it will also pop up, when discussing the implicit version of the *DA*-algorithm.

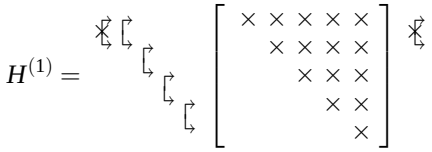
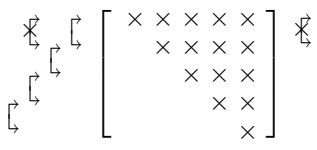
5. Implicit version. The most elegant form for executing steps of the *QR*-algorithm on a Hessenberg matrix performs these steps implicitly. Instead of computing the entire sequence of rotations determining the similarity transformation, the first rotation is sufficient (for a single shifted step) and the remaining rotations are determined on the fly. Applying the similarity transformation determined by this rotation on the matrix perturbs its structure in the upper left corner (*initialization step*). Since the resulting structure after the similarity transformation is known one can apply now a sequence of $n-2$ structure restoring transformations (*chasing steps*) such that the resulting matrix meets the structural constraints. Accomplishing a *QR*-step in this manner does hence not require the computation of the complete unitary matrix determining the similarity transform in advance.

Implicit *QR*-algorithms for Hessenberg(-like) matrices are common knowledge [31, 40, 42, 44]. The most popular method is the bulge chasing method where the elements perturbing the Hessenberg structure are chased downwards untill they slide off the matrix. When exploiting the rotational *QR*-factorization for studying the bulge chasing algorithm, a rotation chasing method is obtained. In this case a perturbing rotation is chased downwards until it vanishes.

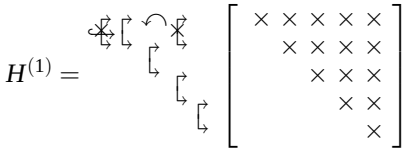
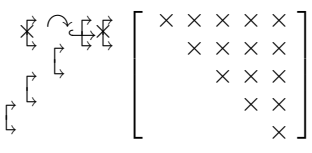
Performing a *DA*-iteration on a zigzag pattern exploits chasing techniques from both ascending (Hessenberg-like) and descending (Hessenberg) sequences. A brief recapitulation of these techniques, followed by a strategy to combine both is considered next and results in an implicit *DA*-iteration.

5.1. Descending and ascending sequences. Only the single shifted case is considered, implying that the initialization step is composed of a single rotation. To pinpoint the difference between descending (Hessenberg) and ascending (Hessenberg-like) sequences both cases are discussed simultaneously. The major difference lies in the determination of the chasing rotations. In the descending case these rotations pop up on the left side, whereas in the ascending case they appear on the right side of the factorization. The initial rotation is determined by from the first column of the Hessenberg matrix $Q_dR_r - \mu Q_a^H$ from (4.5).

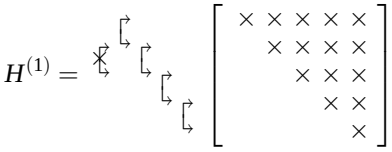
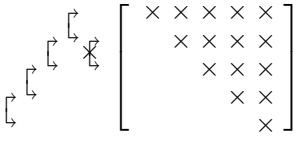
5.1.1. Initialization. Consider a Hessenberg matrix H and Hessenberg-like matrix Z factored as in (2.3). The next graphical scheme depicts the initialization step applied on H and Z, where G_1 and $\tilde{G}_1$ are suitably constructed: $H^{(1)} = G_1^H H G_1$ and $Z^{(1)} = \tilde{G}_1^H Z \tilde{G}_1$. The rotations involved in the similarity transformation are marked by a $\times$.

$H^{(1)} =$  and $Z^{(1)} =$ 

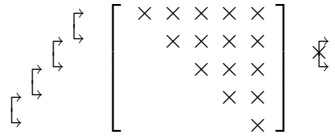
Updating the representation starts by transferring the rotations in both cases on the right through the upper triangular matrix (Lemma 2.6).

$H^{(1)} =$  and $Z^{(1)} =$ 

Removal of one of the two perturbing rotations is done by a fusion on the left in the Hessenberg case and a fusion on the right of the Q-factor in the Hessenberg-like case. After the fusion, a shift-through operation is performed (to the left for the Hessenberg, to the right for the Hessenberg-like), as a result one obtains the following factorizations.

$H^{(1)} =$  and $Z^{(1)} =$ 

To finalize this initialization step we transfer the marked rotation in the middle of $Z^{(1)}$ back to the right.

$Z^{(1)} =$ 

The result of the initialization step gives us for $H^{(1)}$ the QR-factorization of a Hessenberg matrix $H^{(1)} = G_2 \left(Q^{(1)} R^{(1)} \right)$ perturbed on the left with an extra rotation G_2, the QR-factorization of $Z^{(1)} = \left(\tilde{Q}^{(1)} \tilde{R}^{(1)} \right) \tilde{G}_2^H$, however, is the one of a Hessenberg-like matrix perturbed on the right by a rotation $\tilde{G}_2^H$. Both disturbing rotations act on rows 2 and 3.

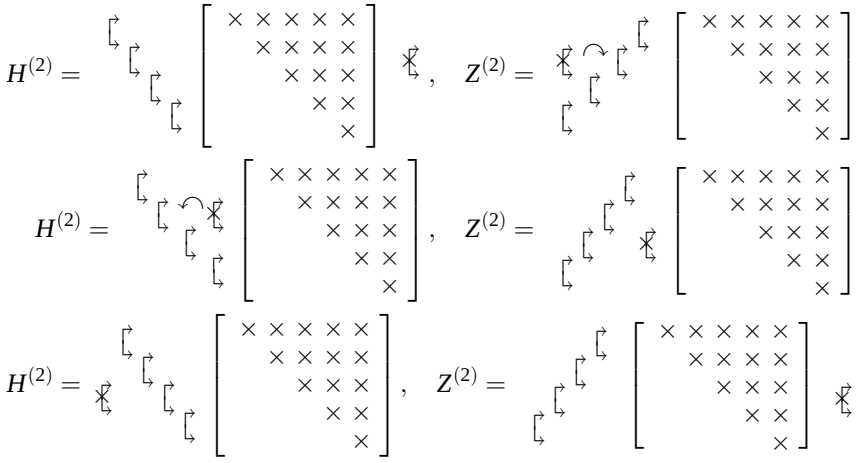
5.1.2. Chasing the rotations. Chasing the rotations in both cases is quite simple. The following two similarity transformations will chase the perturbing rotation down one position.

$$H^{(2)} = G_2^H H^{(1)} G_2 = Q^{(1)} R^{(1)} G_2 \tag{5.1}$$

$$Z^{(2)} = \tilde{G}_2^H Z^{(1)} \tilde{G}_2 = \tilde{G}_2^H \tilde{Q}^{(1)} \tilde{R}^{(1)} \tag{5.2}$$

To deal with $H^{(2)}$, first G_2 is transfered through the matrix $R^{(1)}$ to the left and then a shift through operation is executed resulting again in a matrix factorization $H^{(2)} = G_3 Q^{(2)} R^{(2)}$, where the perturbing rotation G_3 acts now on rows 3 and 4. Dealing with $Z^{(2)}$ proceeds sort of in inverse order. First the shift through operation to the right is performed, followed by transfering the obtained rotation through the upper triangular matrix $\tilde{R}^{(1)}$. The result is a factorization $Z^{(2)} = \tilde{Q}^{(2)} \tilde{R}^{(2)} \tilde{G}_3^H$, with $\tilde{G}_3$ acting on rows 3 and 4. Graphically

these transitions look as follows, starting with the factorizations (5.1) and (5.2) of $H^{(2)}$ and $Z^{(2)}$.



(5.3)

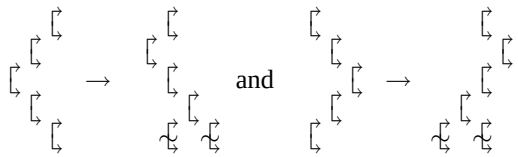
(5.4)

After $n-2$ chasing steps one is not able to perform a shift through operation anymore because the end of the sequence is reached. The perturbing rotation can now be fused with a rotation from the sequence and vanishes. As a result one retrieves again the proper rotational factorizations in a descending or an ascending sequence.

The sequences of rotations considered in the next section are not purely of descending or ascending form. Performing an implicit step on a zigzag pattern will therefore be constituted of both techniques. Executing a similarity transformation to remove an undesired rotation as in the left of Equation 5.4 will be refered to as *removal of a left rotation*, whereas a similarity transformation and the corresponding steps to get rid of the perturbing rotation in the right of Equation 5.3 will be referred to as the *removal of a right rotation*.

5.2. Zizag patterns. The result of a *DA*-iteration step on a *QR*-factorization of a matrix A, where Q is factored in a zigzag pattern results in an upwards shifted zigzag pattern (see Section 4.3). Performing a *DA*-step implicitly is possible and proceeds similarly as in the previous subsection. The remaining issue is, however, the transition from descending to ascending sequences and vice versa. A generic zigzag pattern is constituted of parts which are descending, ascending or turns from ascending to descending and vice versa. The previous section explained purely ascending or descending parts, here focus will be put on the change of direction and on the determination of the final (flexible) rotation. A global implicit version simply combines all the building blocks from the previous and this section.

The upward move of the pattern means that for the connections between ascending and descending sequences the patterns on the left of the arrow in (5.5) turn into the patterns on the right of the arrow. The transitions for both $<$ and $>$ corners are shown in (5.5). The upward move of the pattern leaves the position of the final rotation undetermined, only one of the rotations marked by the $\sim$ sign (5.5) in a sequence can remain.

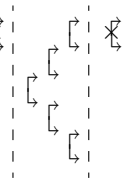
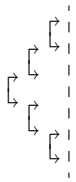
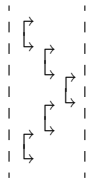
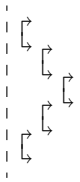


(5.5)

From Subsection 5.1.2 the removal of either a left or right perturbing rotation is known. In the forthcoming graphical flow of the transition from left to right we will omit the upper triangular matrix. The upper triangular matrix does not complicate matters, only the extra transfering of rotations through the matrix should not be forgotten when implementing the method.

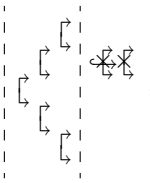
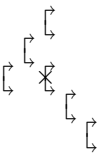
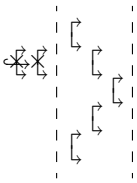
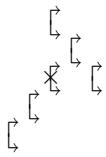
5.2.1. Initialization. The matrix corresponding with $>$-pattern is denoted as $H^{(0)}$ and the matrix $Z^{(0)}$ corresponds to the $<$-pattern. The initial similarity transformation $H^{(1)} = G_1^H H^{(0)} G_1$ and $Z^{(1)} = \tilde{G}_1^H Z^{(0)} \tilde{G}_1$

are carried out just as in Section 5.1, as result we get.

$Z^{(1)} =$  $=$  and $H^{(1)} =$  $=$ 

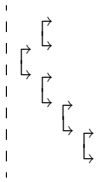
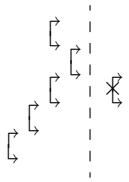
For the matrix $Z^{(1)}$ the disturbing rotation appears on the right, for $H^{(1)}$ it is spotted on the left.

5.2.2. Chasing the rotations. The chasing starts, another time, just as for the purely descending or ascending case. The similarity transformations to obtain $H^{(2)}$ and $Z^{(2)}$ are determined analoguously as before. In Scheme (5.6) the full similarity transformation is depicted, for $Z^{(2)}$ this is based on the removal of right rotation, and for $H^{(2)}$ this is based on the removal of a left rotation.

$Z^{(2)} =$  $=$  and $H^{(2)} =$  $=$  (5.6)

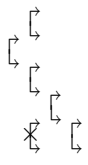
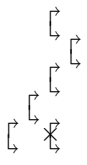
Looking at the resulting schemes in Equation (5.5), we see that after this step the resulting matrix on the right has a change of direction in both patterns. Indeed, after this similarity transformation, we see that the rotations which should determine the upcoming chasing step (marked by $\times$) are not free anymore, which means that they are blocked by other rotations. The change of direction of the corresponding patterns created a sort barrier blocking these rotations.

The problem is easily solved by changing the undesired rotation. Since the blocked rotation satisfies in fact the final pattern, this rotation is kept and the other outer rotations are now marked as undesired.

$Z^{(2)} =$  and $H^{(2)} =$ 

At this point the order of removal is changed. When the first perturbing rotation was on the right (resp. left) it is now on the left (resp. right). These undesired rotations can be removed now by the standard techniques for removing a left or a right perturbing rotation. The process continues by the standard routines for either descending or ascending sequences.

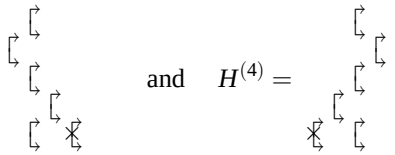
5.2.3. Options for the final rotation. After the initial step and $n-3$ chasing steps the matrices $Z^{(4)}$ and $H^{(4)}$ are of the form (5.7). The rotation marked by $\times$ is the rotation intuitively popping up as the perturbing one after a left or right removal of a rotation by a similarity transformation.

$Z^{(4)} =$  and $H^{(4)} =$  (5.7)

A closer look at the bottom rotations, shows that for both cases, however, the left as well as the right rotation are free. This implies that we can remove either of the bottom rotations. So instead of annihilating

the rotation marked in (5.7), one can also remove the ones marked in (5.8).

$$Z^{(4)} = \qquad \text{and} \qquad H^{(4)} = \tag{5.8}$$



After having removed the undesired rotation in $H^{(4)}$ and $Z^{(4)}$ from either (5.7) or (5.8), the patterns as presented in 5.5 are retrieved. The flexibility with the final rotation was already observed in the study of the structure of *DA*-factorization and also appeared in the explicit version of the *DA*-algorithm.

Based on the implicit *Q*-theorem 3.5 one can prove that the outcome of the implicit *DA*-version is essentially identical to the outcome of the explicit *DA*-version. Both the explicit and the implicit methods applied on A result in unitary similar matrices A_1 and A_2 admitting a *QR*-factorization with an identical pattern for factoring Q, Theorem 3.5 then implies that both matrices are essentially identical.

This approach is clearly related to the *QR*-method. Proofs of convergence exist for both ascending as well as descending sequences. Unfortunately for the moment a proof of convergence for the combination of both is not available. Since the chasing involves a swapping between ascending and descending sequences, its convergence seems somehow natural. In the numerical experiments section it will be shown that the combined approach converges neatly and inherits properties from both techniques.

6. Numerical examples. Due to the almost unlimited choice of possible positions **p** for a compressed unitary matrix it is impossible to test all algorithms and see how they behave numerically for computing eigenvalues. In the upcoming numerical experiments some selected approaches are used. Two types of numerical experiments are executed. In the first type the unitary similarity transformations are examined. The convergence behavior of the eigenvalues in the part already of correct shape (so-called Ritz-values) are studied. The second type of numerical experiments studies the implicit *DA*-algorithm as presented in this paper.

6.1. Software package. Together with this article a software package is provided for studying the patterns and the flexibility of working with the rotational factorization of unitary matrices. The package provides for instance routines for computing the rotational factorization of a unitary matrix; performing a similarity transformation on an arbitrary matrix to obtain a similar *QR*-factored matrix $\hat{A}$ where Q satisfies a previously defined pattern; compute the Q_dQ_a and Q_aQ_d factorization of a compressed unitary matrix; execution of implicit *DA*-steps on a *QR*-factored matrix with Q in compressed form and a tool for visualizing the pattern of a rotational factorization (see Figure 6.1).

In Figure 6.1 patterns for factoring a unitary matrix are shown, visualized with the routine `Gplot`. The left shows the standard piramid factorization of a unitary matrix $Q \in \mathbb{C}^{20\times 20}$. The right figure shows the *CMV*-decomposition of a 20×20 pentadiagonal unitary matrix.

The Matlab software can be downloaded from the author's website at `http://people.cs.kuleuven.be/~raf.vandebril/`.

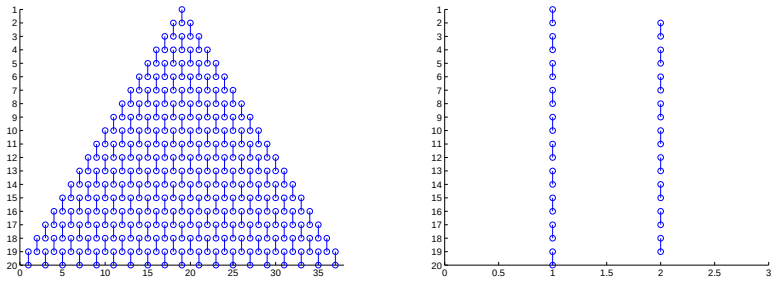


FIGURE 6.1. *The left figure is a graphical piramid shape decomposition of a unitary matrix. The right figure shows the CMV-decomposition of a pentadiagonal unitary matrix.*

6.2. Implementation issues.

6.2.1. Deflation in the *DA*-algorithm. The possibility of subdividing a matrix after a step of the *QR*-method into two or more problems, which can be treated independently is referred to as deflation. In the standard Hessenberg case possible deflation is monitored by checking the relative sizes of the subdiagonal elements. Conveying this technique towards the rotations present in the *QR*-factorization indicates that deflation should be checked by searching for rotations numerically equal to the identity. Indeed, some calculations reveal that a rotation equal to the identity corresponds to a subdivision of the matrix into two submatrices, whose eigenvalues one can now compute independently.

6.2.2. Computational complexity and storage count. Each rotation is stored by two parameters (cosine and the sine), though strictly speaking only a single parameter is sufficient. For each rotation 2 additional integers are needed for determining the order of appearance in the sequence and the row they act on. Hence, globally storing the *QR*-factorization of a matrix $A = QR$, with Q in compressed format uses $1/2n^2 + 9/2n - 6$ parameters.

Important operations in the process are the computation of a Givens rotation, the shift-through operation and a fusion. Computing a Givens rotation takes $c_r = 6$ operations [18], a shift-through operation uses 14 operations plus the computation of 2 rotations implying a cost of $c_s = 14 + 2c_r = 26$ operations. Executing a fusion needs $c_f = 6$ operations.

Assume that the *QR*-factorization of a matrix A is given (roughly $2/3n^3$ operations [18]). Each similarity transformation for annihilation on either the right or the left side uses the same number of operations, so we do not need to distinguish between those. Assume a single left annihilation step is executed on a matrix of size $\ell \times \ell$. This means that $\ell - 2$ rotations are subjected to removal. To remove the top rotation $\ell - 3$ shift-through operations and a single fusion are needed, annihilation of the next rotation uses $\ell - 4$ shift-through operations and a fusion, and this continues. Transition of a rotation through the upper triangular matrix takes $c_t(\ell) = 6(\ell + 1) + c_r = 6(\ell + 2)$ operations. Globally this implies for a left or right annihilation the following cost:

$$c_a(\ell) = \sum_{i=1}^{\ell-2} (\ell - 2 - i)c_s + c_f + c_t = 19\ell^2 - 59\ell + 42.$$

Based on the execution of $n - 2$ annihilation steps of diminishing sizes a global complexity count is retrieved:

$$c_u = \sum_{\ell=1}^{n-2} c_a(\ell) = \frac{19}{3}n^3 - 58n^2 + \frac{515}{3}n - 162.$$

Taking into consideration that the complexity for reducing a matrix to Hessenberg form via rotations takes approximately $5n^3$ operations, one can see that the traditional reduction method is about 25% faster than the alternative version based on the *QR*-factorization.

Consider next an $\ell \times \ell$ matrix on which we want to perform an implicit step of the new *DA*-method. Again, the left and right annihilation steps are equally expensive, so we will focus only on left annihilations. The initial step, computation of the shift and the final step are contributing only to the lower order terms in the complexity count so we will neglect them. Each chasing step involves a transferring through the upper triangular matrix, computation of a rotation and a shift-through operation. So executing $\ell - 2$ chasing steps leads to a complexity of:

$$c_s = (\ell - 2)(c_t(\ell) + c_f) = (\ell - 2)(6\ell + 36).$$

The complexity of a traditional chasing step applied on a Hessenberg matrix takes

$$(\ell - 2)(6\ell + 24),$$

which is just few operations cheaper than the new approach.

Deriving the computational complexity of the global *DA*-procedure is quite cumbersome, since it involves parameters assuming when deflation will take place and so forth. Since all variants are equally expensive we will therefore in the numerical experiments focus on the number of iterations it takes before deflation can be applied. This is one measure one can take for deducing the speed of the approach.

6.3. Ritz-value convergence. When reducing a symmetric matrix to tridiagonal form, the eigenvalues of the part already in tridiagonal shape approximate the eigenvalues of the original matrix. Under some mild assumptions the well-separated eigenvalues are found first [5, 19, 20]. In [29] it was already noticed that the convergence of the part already in Hessenberg-like form equals the convergence of rational Lanczos. Considering the reductions in this work, based on the proof of Theorem 3.5, we see that they are sort of in between the Hessenberg and Hessenberg-like case. The corresponding Krylov spaces are not purely constructed by powers of A or by A^{-1}, but combinations of both. Depending on the choice of the left or right annihilation, successive powers of A or A^{-1} appear in the Krylov subspace. With Ritz-values we refer to the eigenvalues of the part already in compressed format. In the plots of Figure 6.2 the Ritz-value behavior for a symmetric matrix having 100 eigenvalues equally spaced in the interval $[1/n, 1]$ is shown. The horizontal axis denotes the order of the part of the matrix already of desired format. The vertical axis denotes the range of the eigenvalues of the original matrix. In the figures a small dot is depicted if a Ritz-value approximates an eigenvalue in the range $[10^{-2.5}, 10^{-5}]$, a bigger dot is depicted if the approximation lies within $[10^{-5}, 10^{-7.5}]$ and a plus sign is plotted if the approximation is better than $10^{-7.5}$. Six different reduction types are considered, from left to right we have: the reduction to Hessenberg form; the reduction to Hessenberg-like form; the reduction to *CMV*-form; the reduction to a symmetric zigzag-pattern, with 10 annihilations on the left, followed by 10 on the right; the reduction to a nonsymmetric zigzag pattern, with 5 annihilations on the left, followed by 2 on the right; the reduction to a nonsymmetric zigzag pattern with 2 annihilations on the left, followed by 5 on the right. The convergence of the reduction to Hessenberg

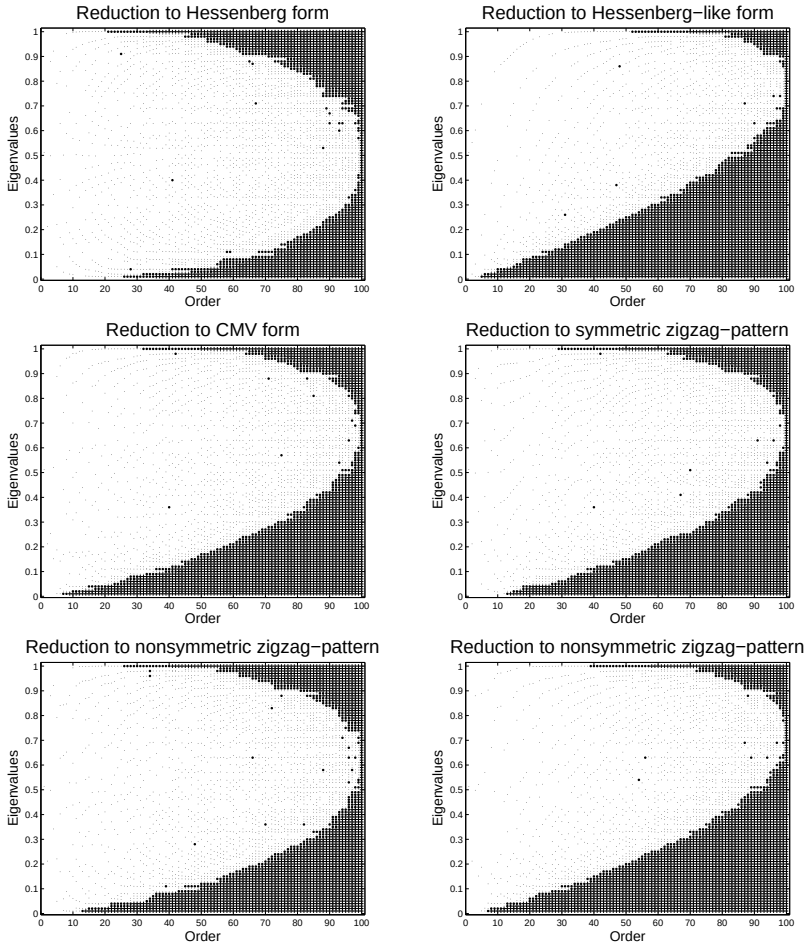


FIGURE 6.2. *Convergence plots of the Ritz-values.*

(tridiagonal) form, shows us the well-known pattern, in which first the well-separated eigenvalues located

on the extremes of the interval are first found. The Hessenberg-like convergence, as is known, is equal to the Ritz-value convergence behavior related to a Krylov space build by A^{-1}. The other figures their convergence is sort of located in between these two figures. For the Hessenberg-like case the convergence is clearly focused towards the smallest eigenvalues, also for the four final plots this is the case, but not as fast since for these matrices also powers of A appear in the sequence. The third and fourth plot are almost identical since they are both symmetric zigzag patterns, the fourth plot alters, however, only after ten steps, whereas the *CMV*-shape alters every step between left and right annihilation. The fourth plot shows thus a slight advantage for convergence close to 1 and a slight delay in convergence to 0. The fifth plot has more powers of A in its pattern than powers of A^{-1}, and the sixth plot vice versa. Clearly the most prominent present term A or A^{-1} forces convergence towards 0 or 1. The fifth picture is closer to the Hessenberg convergence and the sixth picture closer to the Hessenberg-like case.

6.4. Eigenvalue computations. In this section, the accuracy and speed of computing the eigenvalues of an arbitrary matrix are tested. The matrix is first reduced to a compressed *QR*-factorization after which the *DA*-algorithm is used for computing the eigenvalues.

The software provides a generic *DA*-algorithm, able of dealing with any type of pattern of rotations. Only at the end of a chasing the user should specify whether a right or a left annihilation step is executed, with the possibility of changing this during execution. This implementation of the *DA* algorithm is applied on four variants. For these four variants the same input matrix and identical deflation criteria were used making a fair comparison possible. We know that the computational complexity is independent of the type of zigzag pattern. Hence, in the following results only the average number of iterations per eigenvalue is plotted and the maximum relative accuracy.

The initial matrix is reduced via unitary similarity transformations to a compressed *QR*-factorization after which choices for the *DA*-algorithm are made. The four cases are the following:

- Hessenberg matrix, the *DA*-algorithm does not change the pattern at the end of the sequence of rotations, as a result the sequence remains descending throughout the algorithm. This corresponds to the standard Hessenberg algorithm.
- Hessenberg-like matrix, again without changing the pattern at the end of the sequence of rotations in a *DA*-step. This corresponds to the Hessenberg-like algorithm.
- *CMV*-matrix, with altering the final rotation each step. In this way one retains the *CMV*-pattern during the *DA*-algorithm.
- compressed *QR*-factored matrix constructed by two left annihilations and a single right annihilation. The corresponding *DA*-algorithm alters the direction every two steps, so after two *DA*-steps the direction of the trailing rotation changes.

Deflation is applied when the absolute value of the sine of a rotation is below 10^{-14}, this criterion is identical for the four testcases. The algorithm only deflates blocks of sizes at most 2. As shift the Rayleigh shift is taken. The *DA*-method is identical for all four versions, only the position on were the final rotation is taken is different for these four cases.

In the first numerical experiment a random symmetric matrix is generated with eigenvalues equally distributed in the interval $[0,1]$. The problem sizes vary from 20 to 500. The four variants are applied on the same matrix. Figure 6.3 shows the accuracy and the average number of iterations. Taking into consideration that the *DA*-steps are equally expensive for each method, the average number of iterations is a good measure for deducing the speed of the corresponding approach.

All four methods provide almost equally accurate results. Only the number of iterations differs significantly. An inappropriate conclusion states that the Hessenberg-like approach is the best. The *DA*-algorithm is related to a *QR*-method and the convergence of *QR*-methods is dependent on the ratios between eigenvalues. Since the Hessenberg-like approach is related to the inverse of the matrix A, it seems natural that it outperforms the other methods in the case of equals spaced eigenvalues. To illustrate this, a second experiment is performed, with eigenvalues the inverses of the eigenvalues of the first experiment. Figure 6.4 illustrates the results. As expected the Hessenberg case outperforms now the Hessenberg-like case. The two other approaches are somehow in the middle of the figure as their convergence most likely relies on powers of A and A^{-1}.

In the final numerical experiment we changed the *DA*-algorithm and made it adaptive. Instead of selecting the position of the final rotation we alter it depending on the average number of iterations. Starting

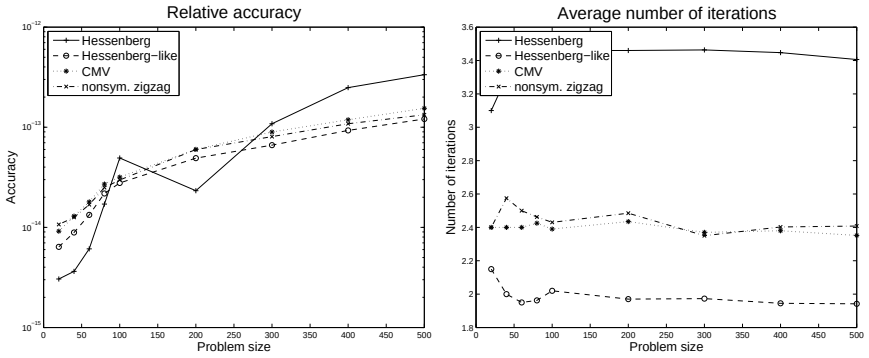


FIGURE 6.3. *Equal spaced eigenvalues.*

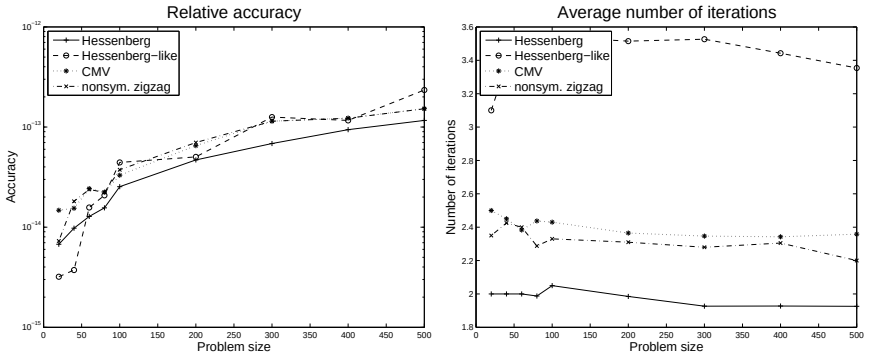


FIGURE 6.4. *Equal spaced eigenvalues.*

e.g. with a Hessenberg matrix, we retain the structure and do not alter the direction of the trailing rotation. It is assumed that 2,5 n *DA*-steps are needed to find all eigenvalues. After 10% of this number, $n/4$ *DA*-steps we check the average number of iterations before convergence occurs. If this is too high we alter the direction. In the Hessenberg case it means that we change the direction of the final rotation, hoping that by performing this action we can reduce the average number of iterations before convergence occurs.

This method was tested on a single example with equal spaced eigenvalues. To compare the flexible approach also the average number of iterations for the Hessenberg and Hessenberg-like approach are shown (Figure 6.5). It is clear that the changing of direction has a significant impact on the average number of

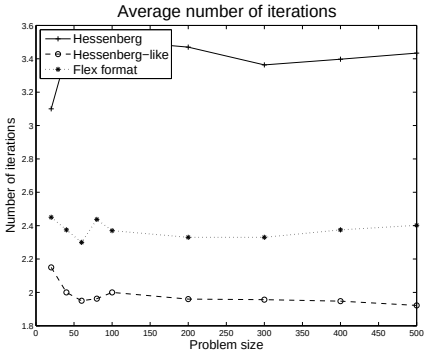


FIGURE 6.5. *Equal spaced eigenvalues.*

iterations and thus also on the global time.

7. Conclusions and future research. In this article a new procedure, based on the *QR*-factorization of a given matrix, is presented for transforming a matrix via unitary similarity transformations to a compressed format. This similarity transformation is a generalization of the reductions to Hessenberg and Hessenberg-like form and it can achieve for instance the *CMV*-pattern in the decomposition of the unitary matrix. Unicity of the reduction is proved. Based on the factorization of the unitary matrix a new procedure, generalizing the *QR*-algorithm, is given. An implicit version of this method is presented. Numerical experiments show the viability of this approach and reveals appealing results related to the convergence of Ritz-values as well as its impact on the convergence of the *DA*-algorithm.

Clearly, the research associated with these algorithms is not yet finished. Important open theoretical questions remain such as a proof of convergence (illustrated by the numerics); an analysis of the convergence speed; a theoretical foundation on which side to choose the trailing rotation; how to pick the shifts,.... These questions will be the subject of further research.

References.

[1] G. S. AMMAR, W. B. GRAGG, AND L. REICHEL, *On the eigenproblem for orthogonal matrices*, in Proceedings of the 25th IEEE Conference on Decision & Control, IEEE, New York, USA, 1986, pp. 1963–1966.

[2] ——, *Constructing a unitary Hessenberg matrix from spectral data*, in Numerical Linear Algebra, Digital Signal Processing and Parallel Algorithms, G. H. Golub and P. Van Dooren, eds., vol. 70 of Computer and Systems Sciences, Springer-Verlag, Berlin, Germany, 1991, pp. 385–395.

[3] G. S. AMMAR AND V. MEHRMANN, *A geometric perspective on condensed forms for hamiltonian matrices*, in Computation and Control II: Proceedings of the Second Bozeman Conference, Bozeman, Montana, August 1-7, 1990, K. L. Bowers and J. Lund, eds., Birkhuser, 1991, pp. 1–11.

[4] R. C. BARROSO AND S. DELVAUX, *Orthogonal Laurent polynomials on the unit circle and snake-shaped matrix factorizations*, J. Approx. Theory, 161 (2009), pp. 65–87.

[5] B. BECKERMANN, S. GÜTTEL, AND R. VANDEBRIL, *On the convergence of rational Ritz-values*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 1740–1774.

[6] D. A. BINI, Y. EIDELMAN, L. GEMIGNANI, AND I. C. GOHBERG, *Fast QR eigenvalue algorithms for Hessenberg matrices which are rank-one perturbations of unitary matrices*, SIAM J. Matrix Anal. Appl., 29 (2007), pp. 566–585.

[7] K. BRAMAN, R. BYERS, AND R. MATHIAS, *The multishift QR algorithm. part I: Maintaining well-focused shifts and level 3 performance*, SIAM J. Matrix Anal. Appl., 23 (2002), pp. 929–947.

[8] ——, *The multishift QR algorithm. part II: Aggressive early deflation*, SIAM J. Matrix Anal. Appl., 23 (2002), pp. 948–973.

[9] A. BUNSE-GERSTNER AND L. ELSNER, *Schur parameter pencils for the solution of the unitary eigenproblem*, Linear Algebra Appl., 154-156 (1991), pp. 741–778.

[10] M. J. CANTERO, L. MORAL, AND L. VELAZQUEZ, *Five-diagonal matrices and zeros of orthogonal polynomials on the unit circle*, Linear Algebra Appl., 362 (2003), pp. 29–56.

[11] S. DELVAUX AND M. VAN BAREL, *The explicit QR-algorithm for rank structured matrices*, Tech. Rep. TW459, Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3000 Leuven (Heverlee), Belgium, May 2006.

[12] ——, *Rank structures preserved by the QR-algorithm: the singular case*, J. Comput. Appl. Math., 189 (2006), pp. 157–178.

[13] ——, *Structures preserved by the QR-algorithm*, J. Comput. Appl. Math., 187 (2006), pp. 29–40.

[14] J. W. DEMMEL, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, Pennsylvania, USA, 1997.

[15] Y. EIDELMAN, L. GEMIGNANI, AND I. C. GOHBERG, *On the fast reduction of a quasiseparable matrix to Hessenberg and tridiagonal forms*, Linear Algebra Appl., 420 (2007), pp. 86–101.

[16] Y. EIDELMAN, I. C. GOHBERG, AND V. OLSHEVSKY, *The QR iteration method for Hermitian quasiseparable matrices of an arbitrary order*, Linear Algebra Appl., 404 (2005), pp. 305–324.

[17] H. FABENDER, *Symplectic methods for the symplectic eigenvalue problem*, Kluwer Academic Publishers, 2002.

[18] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, Johns Hopkins University Press, Baltimore, Maryland, USA, third ed., 1996.

[19] A. B. J. KUIJLAARS, *Which eigenvalues are found by the Lanczos method?*, SIAM J. Matrix Anal. Appl., 22 (2000), pp. 306–321.
[20] ——, *Convergence analysis of Krylov subspace iterations with methods from potential theory*, SIAM Rev., 48 (2006), pp. 3–40.
[21] N. MASTRONARDI, S. CHANDRASEKARAN, AND S. VAN HUFFEL, *Fast and stable two-way algorithm for diagonal plus semi-separable systems of linear equations*, Numer. Linear Algebra Appl., 8 (2001), pp. 7–12.
[22] B. N. PARLETT, *The Symmetric Eigenvalue Problem*, vol. 20 of Classics in Applied Mathematics, SIAM, Philadelphia, Pennsylvania, USA, 1998.
[23] B. SIMON, *CMV matrices: Five years after*, Journal of Computational and Applied Mathematics, 208 (2007), pp. 120–154.
[24] G. W. STEWART, *Matrix Algorithms, Volume II: Eigensystems*, SIAM, Philadelphia, Pennsylvania, USA, 2001.
[25] M. VAN BAREL, R. VANDEBRIL, AND N. MASTRONARDI, *An orthogonal similarity reduction of a matrix into semiseparable form*, SIAM J. Matrix Anal. Appl., 27 (2005), pp. 176–197.
[26] M. VAN BAREL, R. VANDEBRIL, P. VAN DOOREN, AND K. FREDERIX, *Implicit double shift QR-algorithm for companion matrices*, Numer. Math., 116 (2010), pp. 177–212.
[27] R. VANDEBRIL, *The interplay of givens rotations and qr-factorizations*, tech. rep., Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3000 Leuven (Heverlee), Belgium, 2010.
[28] R. VANDEBRIL AND G. M. DEL CORSO, *An implicit multishift QR-algorithm for Hermitian plus low rank matrices*, Tech. Rep. TR-09-06, Dipartemento di Informatica, Università di Pisa, 2009.
[29] ——, *A unification of unitary similarity transformations to compressed representations*, Tech. Rep. TR-10-14, Dipartemento di Informatica, Università di Pisa, 2010.
[30] R. VANDEBRIL, M. VAN BAREL, AND N. MASTRONARDI, *An implicit QR-algorithm for symmetric semiseparable matrices*, Numer. Linear Algebra Appl., 12 (2005), pp. 625–658.
[31] ——, *A new iteration for computing the eigenvalues of semiseparable (plus diagonal) matrices*, Tech. Rep. TW507, Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3000 Leuven (Heverlee), Belgium, Oct. 2007.
[32] ——, *Matrix Computations and Semiseparable Matrices, Volume I: Linear Systems*, Johns Hopkins University Press, Baltimore, Maryland, USA, 2008.
[33] ——, *Matrix Computations and Semiseparable Matrices, Volume II: Eigenvalue and Singular Value Methods*, Johns Hopkins University Press, 2008.
[34] ——, *A parallel QR-factorization/solver of structured rank matrices*, Electron. Trans. Numer. Anal., 30 (2008), pp. 144–167.
[35] ——, *A rational qr-iteration without inversion*, Numer. Math., 110 (2008), pp. 561–575.
[36] D. S. WATKINS, *Understanding the QR algorithm*, SIAM Rev., 24 (1982), pp. 427–440.
[37] ——, *Some perspectives on the eigenvalue problem*, SIAM Rev., 35 (1993), pp. 430–471.
[38] ——, *Bulge exchanges in algorithms of QR-type*, SIAM J. Matrix Anal. Appl., 19 (1998), pp. 1074–1096.
[39] ——, *On the reduction of a Hamiltonian matrix to Hamiltonian Schur form*, Electron. Trans. Numer. Anal., 23 (2006), pp. 141–157.
[40] ——, *The Matrix Eigenvalue Problem: GR and Krylov Subspace Methods*, SIAM, Philadelphia, Pennsylvania, USA, 2007.
[41] ——, *The QR algorithm revisited*, SIAM Rev., 50 (2008), pp. 133–145.
[42] D. S. WATKINS AND L. ELSNER, *Chasing algorithms for the eigenvalue problem*, SIAM J. Matrix Anal. Appl., 12 (1991), pp. 374–384.
[43] ——, *Convergence of algorithms of decomposition type for the eigenvalue problem*, Linear Algebra Appl., 143 (1991), pp. 19–47.
[44] ——, *Theory of decomposition and bulge-chasing algorithms for the generalized eigenvalue problem*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 943–967.
[45] J. H. WILKINSON, *The Algebraic Eigenvalue Problem*, Numerical Mathematics and Scientific Computation, Oxford University Press, New York, USA, 1999.